APSTA-GE 2047 - Fall 2018
Instructor: Ravi Shroff

HOMEWORK ASSIGNMENT 4

Instructions:

- Please submit this assignment on NYU classes by **11:00am on Thursday, 11/15/2018** (the beginning of Week 11 lecture). It is worth 100 pts.
- Work in groups of 2-3, turn in one assignment per group, and indicate group members. *Work together on all parts of the assignment.*
- A submission should include either:
    - a pdf document of all written work and any R scripts you wrote, or
    - one or more RMarkdown files.
- All scripts should be clearly written, commented, and self-contained so that the grader can easily run them to reproduce your analysis.
- You will be graded on completeness, writing and visualization quality, and effort/creativity.
- You have two weeks for this assignment, so please start early, and talk to me if you need help!

1. Scraping Boston Crime Data [50 pts]

In this question, you will scrape and examine Boston crime data from the site Universal Hub. First, you will scrape the crime type and hour from all 20 of the 'Crime by neighborhood' pages on the site, e.g., https://www.universalhub.com/crime/allston.html, https://www.universalhub.com/crime/back-bay.html etc. After that, you will do a quick analysis of the data with some plots.

I suggest you turn in two scripts, crime_analysis.R, and crime_library.R, where crime_library.R loads any libraries, includes any functions, and includes any vectors you reference (e.g., a vector of neighborhood names), and crime_analysis.R loads crime_library.R (with the source() command), and does all parts of the analysis. However, you may include all your work in one script, if you prefer. Turn in a pdf as well with plots and written answers.

A) **[30 pts]** Create a tibble called crime.data which contains *all* crimes in *all* neighborhoods (i.e., each row represents a crime), and has three columns:
    -crime (the name of the crime, from the 'Type' field on each page)
    -hour (the hour as an integer from 0 to 23, from the 'Date' field)
    -neighborhood (the neighborhood name as a string)

B) **[5 pts]** What are the five most common crime types (aggregated across neighborhoods and hours), and how many of each such crime occurred? Be alert for misspellings!

C) **[5 pts]** Make a plot of the total number of crimes (aggregated across neighborhoods and crime types) by hour. Write a few sentences about the pattern you observe.

D) **[5 pts]** Restrict to the five most common crime types, and plot the total number of crimes (aggregated across neighborhoods) for each crime type by hour (i.e., your plot should have five lines). Write a few sentences about the pattern you observe.
E) **[5 pts]** Restrict to just the neighborhoods of Dorchester and Downtown, and plot the total number of crimes (aggregated across crime types (include all crime types, not just the top five)) for each of the two neighborhoods by hour (i.e., your plot should have two lines). Write a few sentences about the pattern you observe.

Hints:

1) I suggest you use or modify the functions in the Lab 8 script UHubScript.R. Specifically, the get_site_content and content_to_parsed_html scripts are useful. You may find it useful to modify extract_all_vict_names2 as well, where you adjust the xpath to extract the 'Type' column and 'Date' column. Recall that you can inspect the source html in most browsers to help figure out how to extract the relevant columns.
2) Make sure to get rid of trailing whitespace and "\n" in the crime type names.
3) Processing the hour is a little tricky. Note that str_split(" - ") from the stringr package will break apart dates and times into a list of lists, and piping this to sapply("[[",2) will return the second element from each of the sub-lists. You may find the parse_date_time() function (along with a %p flag) from the lubridate package useful for extracting the hour as an integer between 0-23.
4) You may find the foreach() and .combine commands used in the stop-and-frisk scripts from HW 3 useful for looping over the 20 urls and combining individual neighborhood tibbles into the crime.data tibble.

2. Characterizing Appeals Courts [50 pts]

In this question, you will build a classifier to examine differences in language between written opinions by two US. federal courts of appeals. Specifically, you will work with a dataset that consists of all opinions from 2010 to 2015 from the 9th circuit court (thought to be the most liberal appeals court) and the 5th circuit court (thought to be among the most conservative appeals courts). Note that there is a large literature in which machine learning techniques—including text analysis—are applied to legal opinions, for instance this, this, and this.

Please turn in one R script called appeals_analysis.R which does all parts of the analysis. Include any written answers in a separate pdf file. Note that we will use both the 'tidytext' and 'tm' libraries (in addition to tidyverse, etc.), so install and load them if necessary.

A) Import the data in the file recent_opinions.tsv as a tibble called 'appeals.data'. Add an 'opinion_id' column to appeals.data, which gives each row a unique id. Load tidytext's stop_words tibble (using the command data(stop_words)), and add the words in custom_words.txt to it to create a custom dictionary of stop words (you can set the lexicon as "custom" for these words). **[5 pts]**

B) You will now build a simple bag-of-words classifier using the top 100 words (that are not stop words) in the corpus.
    a) Unnest the tokens in appeals.data and remove custom stop words. What are the 10 most common words in the entire corpus, and in each of the two circuits (not including stop words)? **[5 pts]**
    b) Build a document-term tibble, where each row represents an opinion (there should be 16389 rows). There should be 102 columns: the circuit (code "fifth" as 1, and "ninth" as 0), the opinion_id, and the number of occurrences of the 100 most common words in the corpus (that are not stop words). For example, if a document contains the word "government" twice, the value in the "government" column for that document would be 2. Randomly shuffle the rows of this tibble, and split into a 50% training set and 50% test set. **[10 pts]**
    c) Fit a logistic regression model on the training set that predicts the circuit as a function of all other predictors. If you got warning messages, what do they say? Compute the AUC of your model on the test set. Explain why your result is strange and which predictor is causing the strange result. **[5 pts]**
    d) Drop the predictor referred to in part c) above, and refit your logistic regression model on the training set. What is your new AUC on the test set? What are the five smallest and five largest coefficients in this model? Give a precise interpretation of the largest model coefficient. **[5 pts]**
C) Repeat sub-parts a), b), and d) of part B) above, but this time, consider the top 100 bigrams instead of individual words. When you are removing stop words, make sure you remove bigrams that contain a stop word as either the first or second word in the bigram. **[5 pts]**
D) Repeat sub-parts b) and d) of part B) above, considering the top 100 bigrams and using the tf-idf value for each of the top 100 bigrams. Compute the tf-idf value for each bigram using the entire corpus of data, not just the training set. **[5 pts]**
E) Suppose you wanted to apply the model you fit in part D) to a single new opinion. Think through how you would do this (write a few sentences about your thoughts). Does part D) actually make sense as a strategy to build a classifier? If not, what is one way you could still use tf-idf values to build a classifier? **[5 pts]**
F) Generate all trigrams, making sure you remove trigrams that contain a stop word as either of the three words in the trigram. Examine, for each circuit, the top 10 trigrams (by frequency in the corpus) that contain the word "supreme." Write a few sentences about what you see (e.g., what are the different contexts in which the 5th vs. 9th circuit opinions are mentioning "supreme"?). **[5 pts]**

Hints:

1) The commands in the tidytext package will do a lot of the work for you. You can adapt commands from Chapter 1 of *Text Mining with R* to quickly unnest tokens and remove stop words. You can adapt commands from Chapter 4 to create bigrams and trigrams (and remove stop words). You can adapt commands from Chapter 5 to create document-term matrices.

2) In part B), subpart b) (and parts C) and D) as well), make sure you include rows for opinions that contain none of the top 100 words/bigrams.
3) When creating the document-term tibbles, you may find the cast_dtm() command helpful. You can pass this command the following flag: weighting = weightTfIdf to automatically create the tf-idf values.
4) Depending on your computer, it may take a little while (but under a few minutes, I think) to apply some of the commands, especially those with bigrams and trigrams.